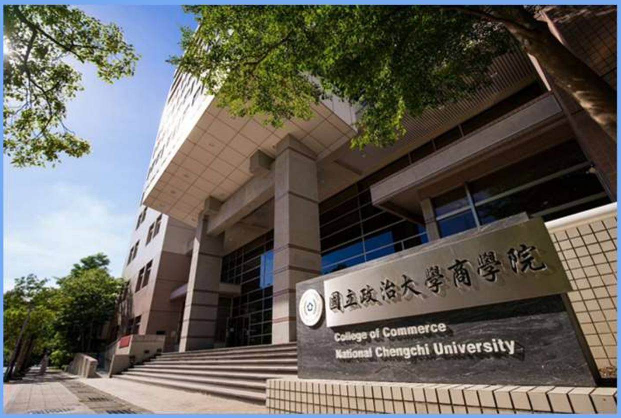
國立政治大學資訊管理學系
NCCU DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

Introduction to Computer Science



Week 4- Operating system II

Shih-Yi (James) Chien
Assistant Professor
Dept. of Management Information Systems
Email: sychien@nccu.edu.tw

Components of OS

Shell is a user interface for access to an OS's services

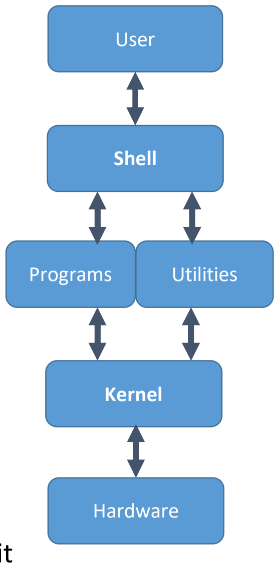
- User give shell a command
- Command-line interface or graphical user interface
- OS can have several different shells

Utilities provide a support process for users

- Common utilities: text editors, search programs, sort programs, etc.

Kernel is the heart of an OS with complete control over everything in the system

- Contain the most basic parts of OS
- Memory management, process management, device/file management
- Other components of the system call on the kernel to perform services
- If the command requires an application, the shell requests kernel to run it



Bootstrap Program

The OS itself needs to be loaded into the memory and run to load other programs into memory for execution

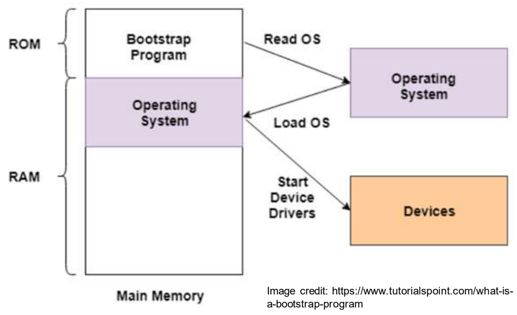
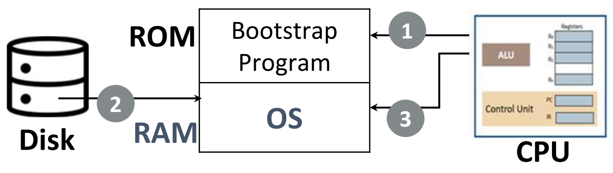


Image credit: https://www.tutorialspoint.com/what-is-a-bootstrap-program

ROM holds a small program called bootstrap

- When the computer is turned on, the CPU counter is set to the first instruction of this bootstrap program and executes the instructions in this program
- Once finished, the program counter is set to the first instruction of the OS in RAM

1. Bootstrap runs
2. OS is loaded
3. OS runs



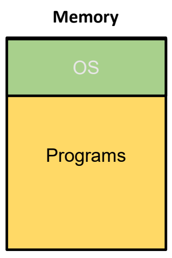
Memory Management

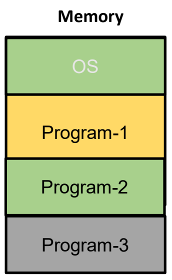
Monoprogramming (belongs to the past)

- Memory capacity is dedicated to a single program
    - Only little part is needed to hold the operating system
- When one program is running, no other program be executed
    - Speed: CPU >> Input & Output
    - CPU is idle when receiving data from or sending data to devices
- If size of program > size of memory: cannot be run

Multiprogramming (current approach)

- Multiple programs are stored in memory and executed concurrently
    - CPU switch rapidly between programs

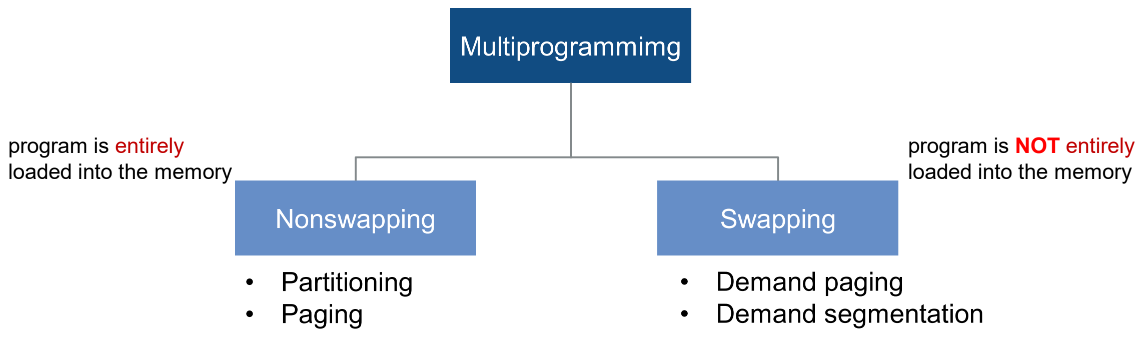




Memory Management- Multiprogramming

Nonswapping: program keeps in memory for the duration of execution

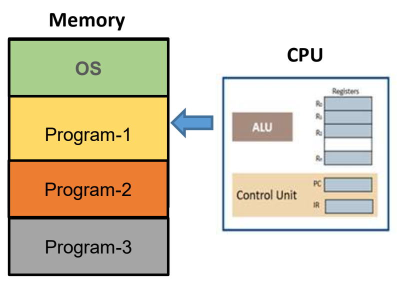
Swapping: programs can be swapped between memory and disk



Nonswapping- Partitioning

Memory is divided into variable-length sections

- Each partition holds one program
- CPU switches between programs
  - Execute instructions of the program, until an I/O operation is encountered or the time allocated for the program expires
- Each program is entirely loaded into the memory, requiring **contiguous** locations
  - Small partition size: programs cannot be loaded into memory
  - Large partition size: holes (unused locations) in memory
  - Memory manager can compact the partitions to remove holes and create new partitions, but it takes extra costs



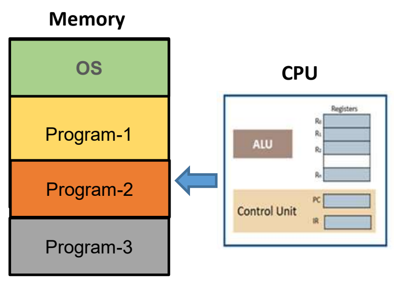


Image credit: Foundations Of Computer Science 4th edition, Behrouz Forouzan, (ISBN:1473751047)

Nonswapping- Paging

Programs are divided into equally sized sections: pages

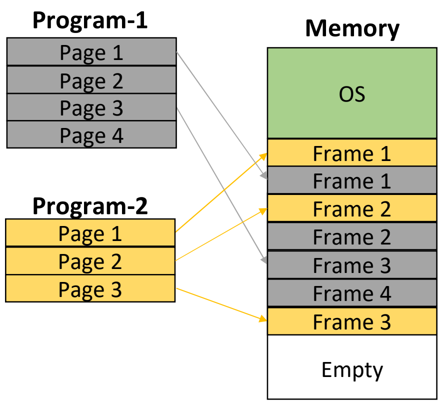
Memory is divided into equally sized sections: frames

- The size of a page/frame is the same and equal to the size of the block used by the system

Programs do not have to be contiguous in memory

- Two consecutive pages can occupy **noncontiguous** frames in memory

Paging can improve efficiency, but the **entire** program needs to be loaded into memory before execution

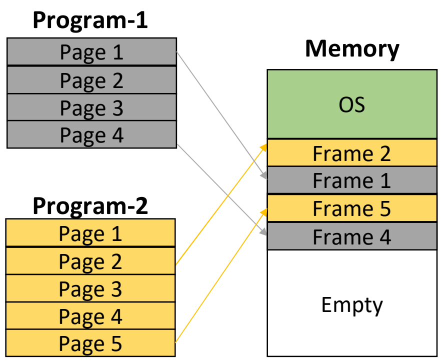


Swapping- Demand Paging

A program is divided into pages, and these pages can be loaded into memory one by one (**not entirely**), and can be executed and replaced by another page

Memory can hold pages from multiple programs at the same time

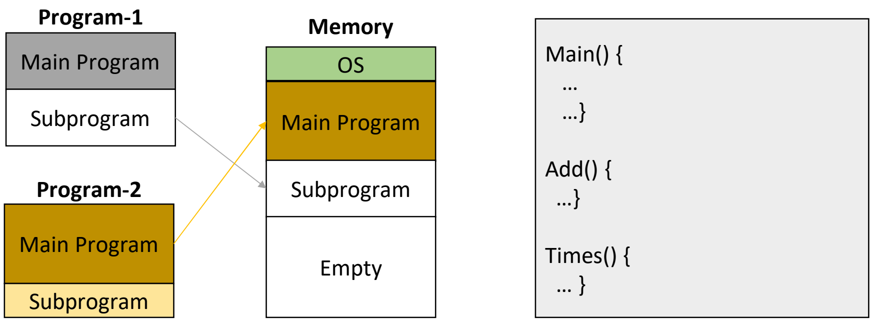
- Pages can be loaded into any free frame



Swapping- Demand Segmentation

A program is usually made up of a main program and subprograms

A program is divided into multiple segments, and the segments are loaded into memory, executed and replaced by another module in the same or different program



```
Main() {
  …
  …}

Add() {
  …}

Times() {
  … }
```

Process Manager

Program: a *nonactive* set of instructions stored on disk

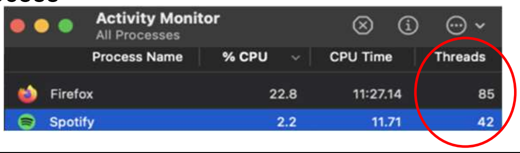
- A program may or may not become a job

Job: a program becomes a job when it is *selected for execution*

- When finished executing, a job becomes a program again
- Every job is a program, but not every program is a job

Process: a program in execution (has started but has not finished)

- A job is being run in memory
- Selected among other waiting jobs and loaded into memory
- Every process is a job, but not every job is a process
- One process can have multiple threads



Activity Monitor
All Processes

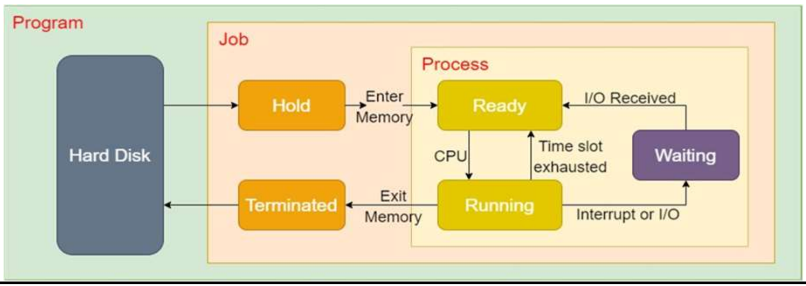
Process Name	% CPU	CPU Time	Threads
Firefox	22.8	11:27.14	85
Spotify	2.2	11.71	42

State Diagram

1. Program becomes Job when selected by OS and bring to Hold state
2. Once being loaded to memory, the Job moves to Ready state and becomes Process
3. When the CPU can execute the Job, it moves to Running state

In Running state, three things can happen:

- Process execution until I/O are needed→ move to Waiting state until I/O is complete
- Process exhausts its allocated time slot → move to Ready state
- Process terminates → move to Terminated state



Schedulers

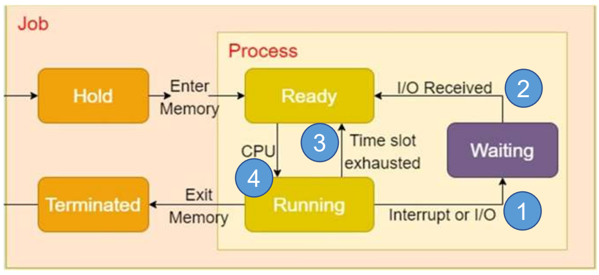
Move a Job or Process from one state to another

Job scheduler

- Create Process from Job: move Job from Hold to Ready state
- Terminate a process: move Job from Running to Terminated state

Process scheduler

1. Move a process: Running → Waiting
   - When the process is waiting for some (I/O) events
2. Move a process: Waiting → Ready
   - When the event is satisfied
3. Move a process: Running → Ready
   - When the process' time allotment has expired
4. Move a process: Ready → Running
   - When the CPU is ready to run the process



Queuing for Scheduling

Process manager uses queues (waiting lists) to store information

- Job/Process control block (PCB): store information about jobs or processes
- Process manager stores the job/process control block
- Job or process remains in memory (can be too large to be saved in a queue)

An OS can have several queues

- Job queue: hold jobs that are waiting for memory
- Ready queue: hold processes that are in memory, ready to be run, and waiting for CPU

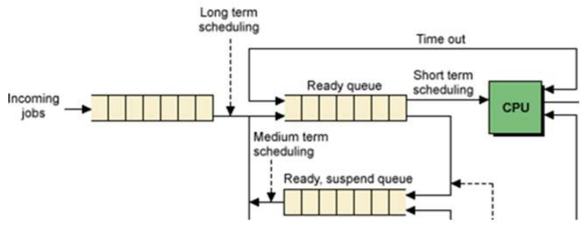


Image credit: https://www.technologyuk.net/computing/computer-software/operating-systems/process-management.shtml

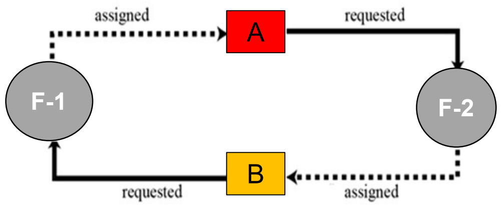
Deadlock

Deadlock occurs when the OS fails to put resource restrictions on processes

- If the OS allows the process to start running without first checking whether the required resources are ready
  - To avoid: cannot start running until the required resources are free
- If the OS allows the process to reserve resources as needed without restrictions
  - To avoid: limit the time a process can hold a resource

When resources are accessed by multiple users

- File-1 is assigned to process-A and cannot release until it acquires File-2
- File-2 is assigned to process-B and cannot release until it acquires File-1



Necessary conditions for Deadlock

The following four conditions are **all necessary** for deadlock to occur :

1. **Mutual exclusion**: only one process can hold a resource (cannot be shared by multiple processes)
2. **Hold and wait**: the process owns a resource, even if it cannot use it before other resources are available (still waiting for resources)
3. **No preemption**: OS cannot temporarily reallocate a resource
4. **Circular waiting**: all processes and resources involved form a loop

Deadlock prevention: preventing at least one of the four required conditions

- Different conditions require different approaches

Starvation

Starvation is the opposite of deadlock

- When OS puts too many resource restrictions on a process

Process-A needs file-1 & file-2

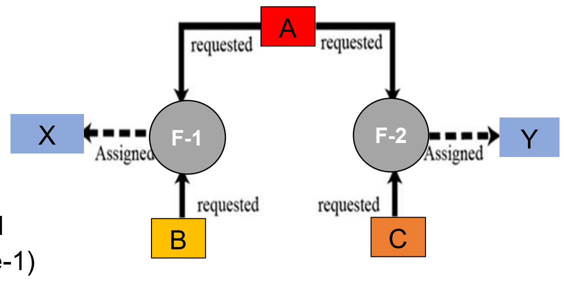
- File-1 is being used by process-X
- File-2 is being used by process-Y

Process-X terminates and release File-1

- Process-A cannot start as File-2 is still occupied
- Process-B is allowed to run (only requested File-1)

Process-Y terminates and release File-2

- Process-A cannot start as File-1 is still occupied
- Process-C is allowed to run (only requested File-2)



CPU Scheduling

A process of determining which process will own CPU for execution while another process is on hold

How to select the next job

- FIFO (first in first out)
- LIFO (last in first out)
- Shortest length first
- Highest priority first

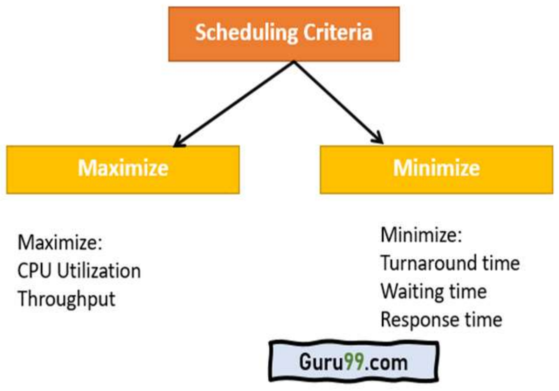


Image credit: https://www.guru99.com/cpu-scheduling-algorithms.html

Scheduling Algorithm- FCFS

First come first serve (FCFS) Scheduling

- Resources utilization in parallel is not possible, which leads to Convoy Effect
  - Convoy effect: a situation where many processes that need to use a resource for short time are blocked by one process holding that resource for a long time
- Non Pre-emptive algorithm: poor resource (CPU, I/O) utilization
- No starvation
  - Starvation is the problem that occurs when high priority processes keep executing and low priority processes get blocked for indefinite time

Scheduling Algorithm- SJF

Shortest-Job-First (SJF) Scheduling

- Works on the process with the shortest burst time or duration first
- The best approach to minimize waiting time and turnaround time
- To successfully implement it, the burst time/duration time of the processes should be known to the processor in advance
- Non Pre-emptive algorithm
- Might lead to starvation

Scheduling Algorithm- RR

Round Robin (RR) Scheduling

- Designed for time-sharing systems
- A fixed time is allotted to each process for execution
  - Quantum or slice
- Once a process is executed for the given time period that process is preempted and another process executes for the given time period
- No starvation
- Pre-emptive algorithm

Device Manager

Device manager is responsible for access to input/output devices

- Monitor every input/output device constantly to ensure that the device is functioning properly
- When a device is ready to serve the next process in the queue
  - Device manager can maintain one or more queues for input/output devices
    - Two queues for two printers
- Control different policies for accessing input/output devices
  - FIFO for one device and LIFO for another

System types

1. Batch system
2. Multiprogramming system
3. Time-sharing system
4. Multiprocessors system
5. Distributed system
6. Real-time system
7. Handheld (mobile) system

Batch system

Designed in 1950s for mainframe computers (large machines)

- No direct interaction b/t a user and a computer

The user/programmer sends a job (written on punched cards) to the operating room

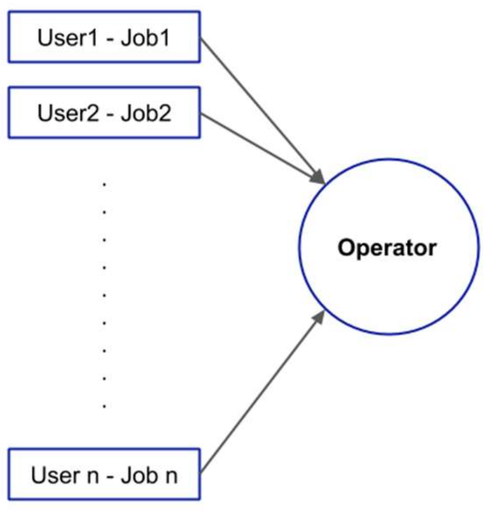
- Cards were fed into the computer by an operator

Jobs are batched together by the required resources

OS only ensures all computer resources transferred from one job to another (increase resources utilization)

- Job control program

Single Task System



Batch System

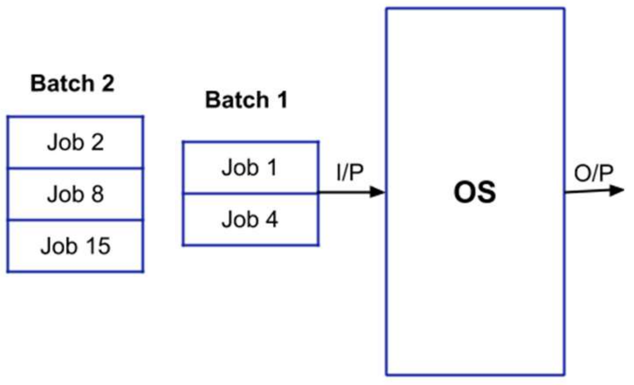


Image credit: https://afteracademy.com/blog/what-are-the-types-of-an-operating-system

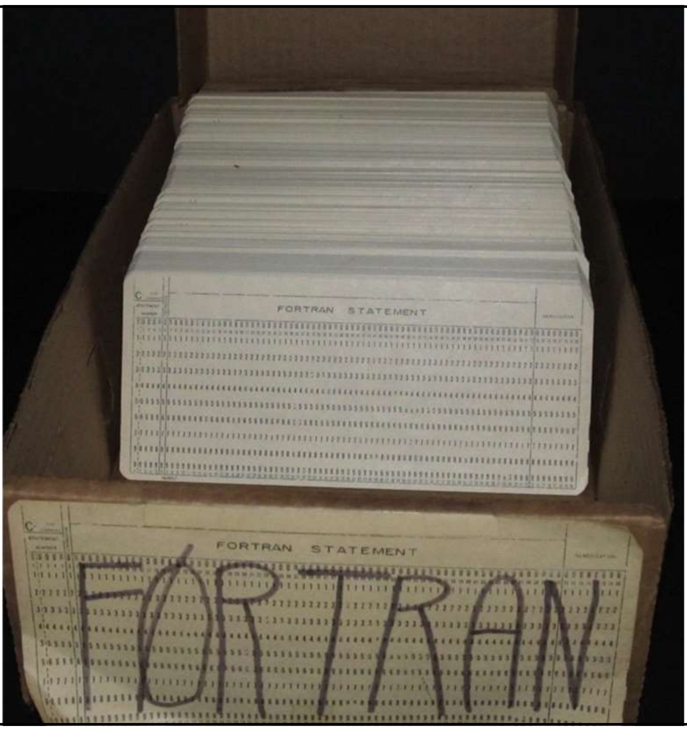


Image Credit: https://www.are.na/block/1023150

Multiprogramming system

To maximize CPU utilization, multiple programs can be loaded into memory for getting to execute

When one program is waiting for I/O transfer, CPU will switch to another program

- CPU's time can be shared into various processes

Multiprogramming degree: the number of processes present in the main memory at any point of time

- Higher multiprogramming degree → higher CPU utilization

Time-sharing system

An extension of multiprogramming system

Allowing many users to interact simultaneously with the computer

Sharing a computing resource among many users at the same time

Time slicing or round-robin scheduling gives all processes an equal opportunity to use CPU

CPU time is divided into slices to be allocated to all processes

- A quantum (its allowance of CPU time) is usually around 100 ms
- Too long: poor response time
- Too short: too many process switches and will lower CPU efficiency

Multiprocessors system

A system has more than one processor (CPU)

- Controlled by the same OS, sharing memory, bus, clocks, and I/O devices
- Enable parallel processing

Asymmetric Multiprocessing (ASMP)

- Master-slave relationship: not all of the CPUs are treated equally
- Master processor controls the entire system (no shared memory)

Symmetric Multiprocessing (SMP)

- No master-slave relationship, two/more identical processors are connected
- Shared main memory, have full access to all I/O devices

Distributed system

Various components are spread across multiple computers on a network

Communicate and coordinate their actions by passing messages to one another from any system

- Connected by network (LAN or WAN)
- Controlled by different OS
- Does not share memory, bus, or I/O devices

Real-time system

Hard real-time OS

- Processing must be done within the defined constraints or the system will fail
- Examples : nuclear systems, a large number of defense applications, avionics

Soft real-time OS

- Can miss some deadlines, but eventually performance will degrade if too many are missed
- Examples : personal computer, audio and video systems

Handheld (Mobile) System

Focused on the needs of a mobile user and the capabilities of the device

- Android
- iOS

Work especially well with mobile device features

- Touchscreens, voice recognition, wi-fi networks

Run with limited memory of mobile devices and the display works with smaller screen size

Types of cloud services: IaaS, PaaS and SaaS

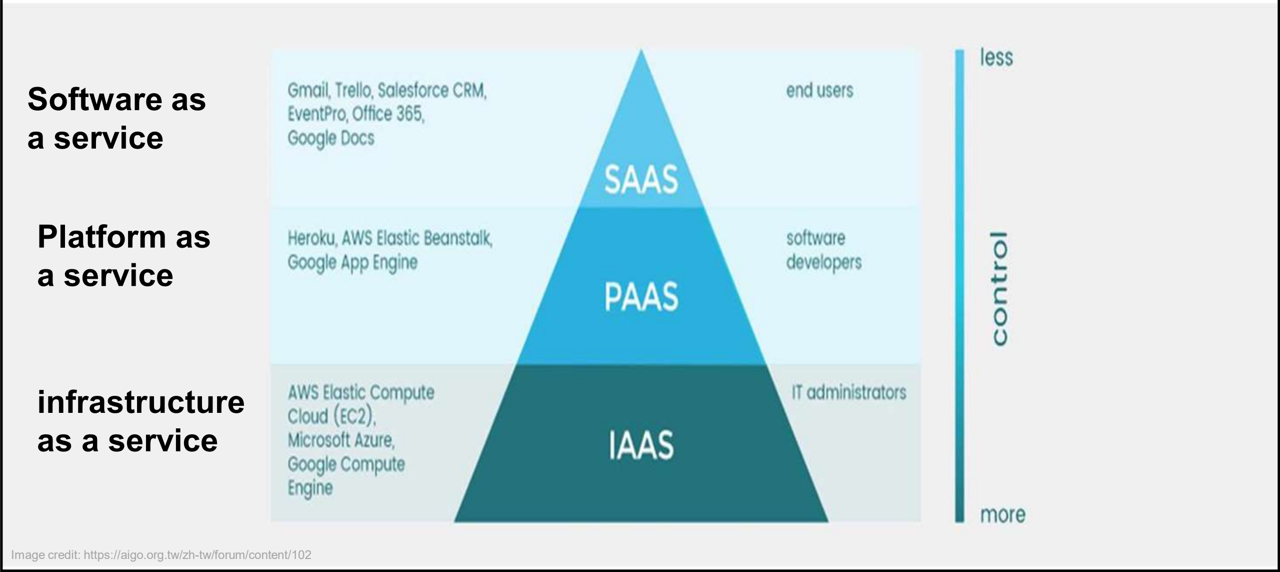
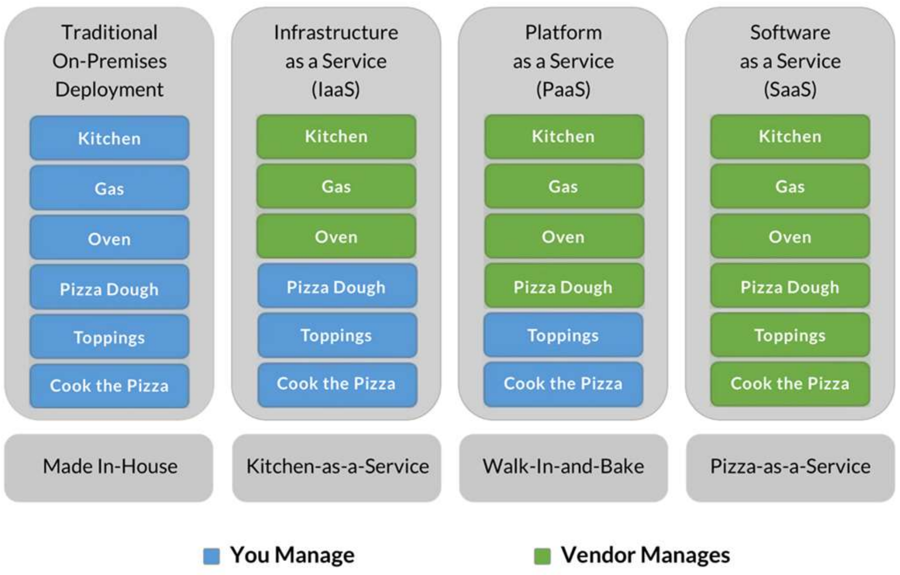

Software as a service
Platform as a service
infrastructure as a service
Gmail, Trello, Salesforce CRM, EventPro, Office 365, Google Docs
Heroku, AWS Elastic Beanstalk, Google App Engine
AWS Elastic Compute Cloud (EC2), Microsoft Azure, Google Compute Engine
SAAS
PAAS
IAAS
end users
software developers
IT administrators
less
control
more

Image credit: https://aigo.org.tw/zh-tw/forum/content/102

New Pizza as a Service

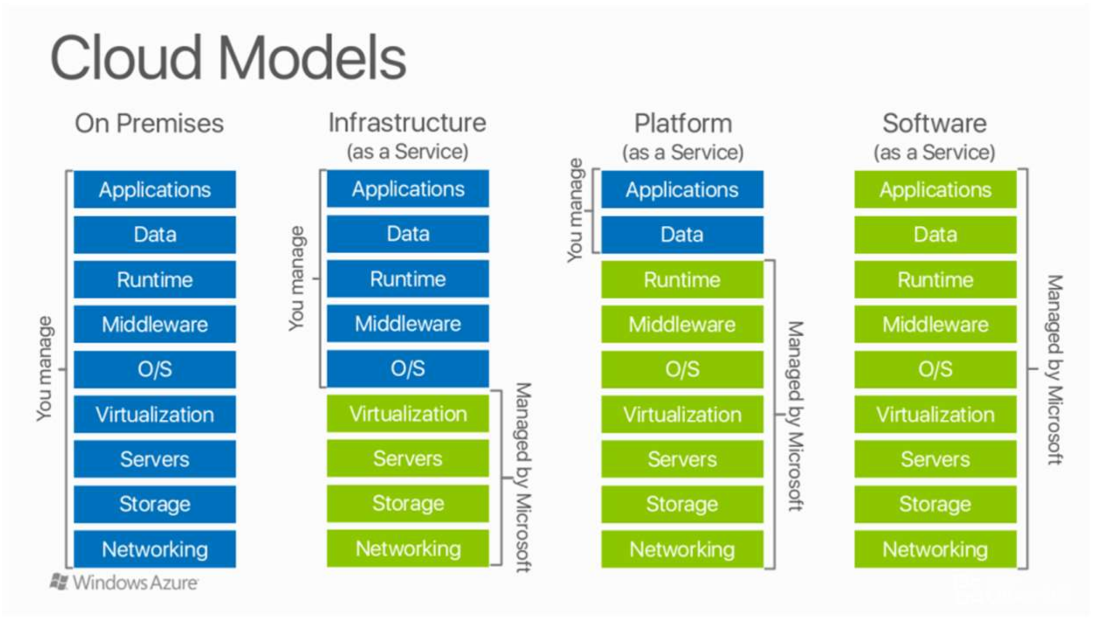


Traditional On-Premises Deployment	Infrastructure as a Service (IaaS)	Platform as a Service (PaaS)	Software as a Service (SaaS)
🟦 Kitchen	🟩 Kitchen	🟩 Kitchen	🟩 Kitchen
🟦 Gas	🟩 Gas	🟩 Gas	🟩 Gas
🟦 Oven	🟩 Oven	🟩 Oven	🟩 Oven
🟦 Pizza Dough	🟦 Pizza Dough	🟩 Pizza Dough	🟩 Pizza Dough
🟦 Toppings	🟦 Toppings	🟦 Toppings	🟩 Toppings
🟦 Cook the Pizza	🟦 Cook the Pizza	🟦 Cook the Pizza	🟩 Cook the Pizza
Made In-House	Kitchen-as-a-Service	Walk-In-and-Bake	Pizza-as-a-Service

🟦 **You Manage** 🟩 **Vendor Manages**

Image crdit: https://m.oursky.com/saas-paas-and-iaas-explained-in-one-graphic-d56c3e6f4606

Cloud Models



On Premises	Infrastructure (as a Service)	Platform (as a Service)	Software (as a Service)
🟦 Applications	🟦 Applications	🟦 Applications	🟩 Applications
🟦 Data	🟦 Data	🟦 Data	🟩 Data
🟦 Runtime	🟦 Runtime	🟩 Runtime	🟩 Runtime
🟦 Middleware	🟦 Middleware	🟩 Middleware	🟩 Middleware
🟦 O/S	🟦 O/S	🟩 O/S	🟩 O/S
🟦 Virtualization	🟩 Virtualization	🟩 Virtualization	🟩 Virtualization
🟦 Servers	🟩 Servers	🟩 Servers	🟩 Servers
🟦 Storage	🟩 Storage	🟩 Storage	🟩 Storage
🟦 Networking	🟩 Networking	🟩 Networking	🟩 Networking

Image credit: https://techcommunity.microsoft.com/t5/azure-developer-community-blog/transitioning-from-on-prem-virtual-machines-to-more-cost/ba-p/336595